REMARKS

This communication is being filed in response to the final Office Action having a mailing date of December 27, 2006, in which a three (3) month Shortened Statutory Period for Response has been set, due to expire March 27, 2007. Nineteen (19) claims, including three (3) independent claims, were paid for in the application. Claim 6 is currently amended. New claims 20-24 are added. No new matter has been added to the application, and all claims are now in condition for allowance. No fee for additional claims is due by way of this Amendment. The Director is authorized to charge any additional fees due by way of this Amendment, or credit any overpayment, to our Deposit Account No. 19-1090. Upon entry of the amendments herewith, claims 1- 24 remain pending.

I. Rejections under 35 U.S.C. § 103

At sections 2 through 21 of the Office Action (pages 2-14), claims 1-19 were rejected under 35 U.S.C. § 103(a) as allegedly unpatentable over ***Padmanabhan et al.*, (U.S. Patent Application 2004/0221141)**, hereinafter *Padmanabhan*, in view of ***Christie et al.*, (WO 02/13005)**, hereinafter *Christie*, in further view of ***Pilat et al.*, (U.S. Patent 4,448,173)**, hereinafter *Pilat*, and ***Shaylor et al.*, (U.S. Patent 6,408,325)**, hereinafter *Shaylor*.

Applicants disagree with the above stated rejections. It is well established at law that, for a proper rejection of a claim under 35 U.S.C. §103(a) as being obvious based upon a combination of references, the cited combination of references must disclose, teach, or suggest, either implicitly or explicitly, all elements and/or features of the claim at issue. See, *e.g.*, *In Re Dow Chemical*, 5 U.S.P.Q.2d 1529, 1531 (Fed. Cir. 1988), and *In re Keller*, 208 U.S.P.Q. 871, 881 (C.C.P.A. 1981). Applicants believe that all amended claims are patentable and that all the remaining claims are also patentable.

A. Independent Claims 1 and 9

Applicants respectfully submit that independent claims 1 and 9 are allowable for at least the reason that the proposed combination of *Padmanabhan* in view of *Christie* and in

further view of *Pilat* and *Shaylor* does not disclose, teach, or suggest at least the feature of "a variable number of registers that varies according to the value of at least one flag stored in a register to be saved" as recited in the claims (emphasis added).

a. *Padmanabhan* does not disclose a variable number of registers

In the "Description of the Related Art" section of the present application (*Padmanabhan*), the applicants disclose a microprocessor, memory array having stack space for contextual data and a stack pointer, a central processing unit having registers with contextual data, and the storage of contextual data upon a switch from a first program to a second program. The applicants agree with Examiner's notation of this prior art at section 4 of the Office Action (pages 2-3), and the applicants further agree that *Padmanabhan* fails to teach storage in a variable number of registers according to the value of at least one flag stored in a register to be saved.

b. *Christie's* flags indicate whether an extended register set is enabled, but not how many registers should be saved

Christie relates to a microprocessor of the x86 family having a "32 bit compatibility" mode, which executes 32 bit operands, and a "32 bit register extension" mode ("REX32"), which executes 64 bit operands. Page 5, lines 33-37. *Christie's* CPU 32 comprises a REX32 Bit 63 flag and an RX Bit 66 flag, which allow configuration of *Christie's* operating mode globally or on a process-specific basis. Page 6, lines 1-17; Fig. 5. The global enable REX32 Bit 63 flag can only be set by "privileged code such as an operating system, a basic input output system or BIOS, or a supervisor-mode utility," but an application program currently being executed can set the RX Bit 66 flag. Page 7, lines 1-4. The REX32 mode can only be entered when both REX32 Bit 63 and RX Bit 66 are set. Page 6, line 16. Accordingly, it is *Christie's* flags that indicate when the extended register set is enabled.

However, when discussing his context switch data structure, *Christie* expressly recites that "*a problem may arise in implementing extended register set 86 of register file 60 in that a 32 bit operating system is most likely unaware of the extended registers and may thus*

make no attempt to save the extended registers during context switches." Page 10, lines 37-40. This important caveat is repeated two more times on Page 11 at lines 7–8 and lines 18-20. One skilled in the art will recognize that if the operating system is unaware of the extended registers, it is also unaware of the bit flags that indicate the operating mode and thus may not use the information to save "a variable number of registers that varies according to the value of at least one flag" or a flag set "such that the number of registers that store contextual data is variable." Accordingly, independent claims 1 and 9 are allowable.

c. *Christie's* flags are not stored in a register to be saved

Christie's global enable REX32 Bit 63 and process enable RX Bit 66, which indicate when the extended register set is enabled, may be stored in Control Register 62 and Flags Register 64 respectively. Page 3, lines 5-19. However, *Christie* further discloses which of his registers are saved during a context switch, and Control Register 62 and Flags Register 64 are not included. Instead, during a context switch, *Christie* saves Register File 60, which is shown in Figs 5, 7, and 10. Register File 60 has a Standard Register Set 84 and an Extended Register Set 86. *Christie's* Fig. 11 shows the 512-byte structure of the memory array reserved for saving Register File 60 during context switches. The 512-byte structure comprises two parts 122 and 124:

Part 122 receives the context of the 32-bit operating mode and includes general-purpose registers (EAX, EBX, ECX, EDX, ESP, ESP, EBP, ESI, and EDI), standard MMX registers, and standard SSE registers. Page 11, lines 14-26 and 33-36.

Part 124 receives the additional context of the extended operating mode REX32 and includes extended portions of SSE, MMX, and SSE registers. Page 11, lines 32-36.

Nowhere does *Christie* disclose, teach, or suggest that his Control Register 62 and Flags Register 64 (including global enable REX32 Bit 63 and process enable RX Bit 66) "are stored in a register to be saved" during a context switch. Accordingly, claims 1 and 9, which recite "a variable number of registers that varies according to the value of at least one flag stored in a register to be saved" are allowable. Further, claim 17, which recites, "a memory array having stored therein contextual data" and a first group of the registers storing contextual data

and ... the flag is stored in a register to be saved as part of the program contextual data," is also allowable.

d. *Christie* teaches away from flags stored in a register to be saved

In contrast to storing his flags in a register to be saved, *Christie* expressly teaches away from saving global enable REX32 Bit 63 and process enable RX Bit 66 during a context switch. As clearly seen in *Christie's* Fig. 5, Control Register 62 and Flags Register 64 are part of the Decode Control Unit 58, and are not part of the Register File 60. As described above, it is the registers in the Register File 60 that are saved during a context switch and not the registers in the Decode Control Unit 58. Along these lines, it is also apparent to one skilled in the art that registers in a decode control unit are classically controlled by the microprocessor's BIOS and are not part of the context. For at least the reason that *Christie* teaches away from storing his flags in a register to be saved, claims 1 and 9 are further allowable.

e. *Pilat* sets his flag differently than the present claims

The addition of the *Pilat* reference does not cure the deficiencies of *Padmanabhan* and *Christie*. *Pilat* discloses a system in which variable amounts of state exist and that it would be wasteful to store excess data for operations that do not require said data, but *Pilat* teaches a mechanism different than what is claimed herein. *Pilat* uses specific call instructions to determine whether both basic state and extended state will be saved on the stack or only basic state. Col. 5, lines 19-29. More specifically, *Pilat's* call instruction reveals the quantity, and based on the call, *Pilat* sets his "frame type specifier" flag to determine how much information *was* saved. Col. 39, lines 12-13. To the contrary, present claims 1 and 9 recite "a variable number of registers that varies according to the value of at least one flag," and present claim 17 recites "a second group of the registers not storing contextual data when a flag has a first value and switching to store contextual data also in the second group of registers when the flag switches to a second value, such that the number of registers that store contextual data is variable." *E.g.*, the present claims look forward while the cited reference looks back. This distinction is further reason why none of the cited references, singly or in any motivated

combination, teach the elements recited in the present claims. Independent claims 1 and 9 are therefore allowable, and the Applicants respectfully request that the rejections be withdrawn.

f. *Shaylor* does not add elements relevant to the present claims

The *Shaylor* reference is not relied on by the Examiner, but was mentioned in the context of "extrinsic evidence." *Shaylor* is directed to the field of processors with very long instruction word (VLIW) architectures and large register files; it is not well related to the present claims. Additionally, *Shaylor* individually tracks registers and marks bits inside each register as "dirty" and "valid," which are also not related to the present claims. For the reason that *Shaylor* does not add evidence of obviousness, the applicants respectfully request withdrawal of any rejection relying on this reference.

B. Independent Claim 17

Applicants respectfully submit that independent claim 17 is allowable along the same lines as independent claims 1 and 9. As discussed above, *Christie's* operating system is unaware of the extended registers, and therefore unaware of the extended register bit flags. Accordingly, *Christie* does not disclose, teach, or suggest at least the feature of "the number of registers that store contextual data is variable ... and ... contextual data to be stored in the first group only or in both the second group and the first group, [is] based on the flag value" as recited in claim 17 (emphasis added). *Christie* does not store his Control Register 62 and Flags Register 64 during a context switch, so claim 17, which recites, "a memory array having stored therein contextual data" and a first group of the registers storing contextual data and ... the flag is stored in a register to be saved as part of the program contextual data," is also allowable.

With further regard to claim 17, the addition of *Pilat* and *Shaylor* does not cure the deficiencies of *Padmanabhan* and *Christie*. As stated above, *Pilat's* special call instructions determine how much data will be saved, and in contrast, claim 17 recites "a second group of the registers not storing contextual data when a flag has a first value and switching to store contextual data also in the second group of registers when the flag switches to a second value, such that the number of registers that store contextual data is variable." As also stated above,

Shaylor is not relied on by the Examiner or well related to claim 17. For each of the reasons stated above, claim 17 is in condition for allowance.

C. Dependent Claims 2 and 10

Dependent claims 2 and 10 expressly recite, "changing the value of the flag according to the content of a register." To the contrary, there is no disclosure, teaching, or suggestion in *Christie* that the value of a flag is changed according to the content of a register. Instead, *Christie* utilizes a change in the microcode underlying specific FXSAVE and FXRSTOR instructions to store and retrieve his register set. Page 11, lines 14-32. Accordingly, applicants respectfully request an indication of allowability of dependent claims 2 and 10.

D. Dependent Claims 3 and 11

Dependent claims 3 and 11 expressly recite, "changing the value of the flag according to the content of an extended addressing register of a program counter of the central processing unit." For at least two reasons, there is no disclosure, teaching, or suggestion in *Christie* that the value of his flags are changed according to the content of an extended addressing register of a program counter.

First, *Christie's* REX32 Bit 63 and RX Bit 66 are the flags that allow configuration of *Christie's* operating mode globally or on a process-specific basis. Page 6, lines 1-17; Fig. 5. The global enable REX32 Bit 63 flag is set by "privileged code such as an operating system, a basic input output system or BIOS, or a supervisor-mode utility," and the RX Bit 66 flag is set by an application program currently being executed. Page 7, lines 1-4. It is thus clear that neither of *Christie's* flags is set "according to the content of an extended addressing register."

Second, *Christie* does not disclose a program counter (or 'instruction pointer' as per processors of x86 architecture) or equivalent register in his specification. Instead, as shown in Fig. 5, REX32 Bit 63 and RX Bit 66 are formed in Control Reg 62 and Flags Reg 64, not in "an extended addressing register of a program counter." For at least the reasons that there is no teaching from *Christie* regarding "changing the value of the flag according to the content of an

extended addressing register of a program counter of the central processing unit," dependent claims 3 and 11 are allowable.

E. Dependent Claims 4 and 12

Dependent claims 4 and 12 expressly recite, "when the content of the extended addressing register is equal to 0" and "when the content of the extended addressing register is not equal to 0." However, the rejection of claims 4 and 12, recites at sections 7 and 14 (pages 5-6 and 9-10 of the Office Action) that "While the specific teachings of what happens when the content is equal to 0 is not taught by the references...". As stated above, a proper rejection under 35 U.S.C. §103(a) requires that references disclose, teach, or suggest all elements and/or features of the claim at issue. See, *e.g.*, *In Re Dow Chemical* and *In re Keller*. For at least the reasons that none of the references, singly or in any motivated combination, disclose, teach, or suggest all of the elements of dependent claims 4 and 12, the applicants respectfully request withdrawal of the rejection.

F. Dependent Claims 7 and 15

Dependent claims 7 and 15 expressly recite, "saving the register containing the flag last." In contrast to the present claims, there is no disclosure, teaching, or suggestion in *Christie* regarding the order in which registers are saved. Instead, *Christie* teaches away from the need to save registers in any particular order because *Christie* does not save a variable number of registers. Accordingly, dependent claims 7 and 15 are allowable.

II. Dependent Claims in General

Each dependent claim inherits the limitations of its respective base claim and all intervening claims. Therefore, allowance of the respective base claim compels allowance of all dependent claims. Accordingly, dependent claims that were referenced in the Office Action and not specifically referenced in the present response are allowable for at least reasons of their respective base claims.

III. New Claims 20-24

New independent claim 20 is allowable for at least the reason that the cited references do not have the element of a central processing unit arranged to "conditionally store a variable number of registers of the second group of registers." With respect to the cited references, it has already been established that *Christie* will unconditionally store all registers, basic and extended, by modification of the microcode for specific instructions. Page 11, lines 14-32. Similarly, *Pilat* discloses his call procedure in detail beginning at Col. 35, line 27 and ending at Col. 43, line 33. In the procedure, *Pilat* discloses hard-coded call instructions to unconditionally save a fixed number of registers that will be saved based on the instruction. See, for example, Col. 35, lines 57-58. The fixed number of registers saved for both an N-Call Frame 1239 and a G Call Frame 1241 are shown in *Pilat's* Fig. 12. Independent claim 20 is thus allowable, and dependent claims 21-24, which depend therefrom are also allowable.

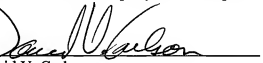
IV. Conclusion

In light of the above amendments and remarks, Applicants respectfully submit that all objections and/or rejections have been traversed, rendered moot, and/or accommodated, and that all pending claims 1-24 are allowable. Applicants, therefore, respectfully request that the Examiner reconsider this application and timely allow all pending claims. The Examiner is encouraged to contact Mr. Carlson by telephone to discuss the above and any other distinctions between the claims and the applied references, if desired.

No fee for additional claims is due by way of this Amendment. The Director is authorized to charge any additional fees due by way of this Amendment, or credit any overpayment, to our Deposit Account No. 19-1090.

Respectfully submitted,

SEED Intellectual Property Law Group PLLC



David V. Carlson
Registration No. 31,153

DVC:jr

701 Fifth Avenue, Suite 5400
Seattle, Washington 98104
Phone: (206) 622-4900
Fax: (206) 682-6031

852663.405 / 890246_1.DOC